Pergamon

Computers Math. Applic. Vol. 35, No. 7, pp. 117–128, 1998
Published by Elsevier Science Ltd
Printed in Great Britain
0898-1221/98 $19.00 + 0.00

PII: S0898-1221(98)00039-X

A Scalable Model for Complex Flows

O. YAŞAR
Center for Computational Sciences
Oak Ridge National Laboratory
P.O. Box 2008, MS-6203
Oak Ridge, TN 37831, U.S.A.
yasar@ccs.ornl.gov

Abstract—We describe a scalable parallel algorithm for numerical simulations of turbulent, radiative, magnetized, and reactive fluid + particle systems on message-passing, distributed-memory computers. Accurate simulation of such complex flows has applications in engine combustion, industrial pulverized coal burners, astrophysics, inertial confinement fusion, nuclear systems, and many other strategically and economically important areas. Our algorithm has been developed based on a widely-used combustion code KIVA-3, a plasma and radiation hydrodynamics code R-MHD, a classical particle dynamics code CMDT, and a discrete ordinates particle transport code TORT. The development is being done on the Intel Paragon with PVM and MPI extensions. We report high levels of parallel efficiency and scalability (up to 1024 nodes) for a baseline engine test case, using our current message-passing reactive and turbulent flow code. The three-dimensional extension of radiation magnetohydrodynamics component is still being worked at and we hope to report further progress in the future.

Keywords—Combustion, Particle dynamics, Turbulence, Radiative transfer, Magnetohydrodynamics, Spark ignition, Massively parallel computing.

1. INTRODUCTION

The phenomenon of compressible flow is of fundamental importance in a wide variety of contexts. It plays a role in the wakes and boundary layers of aircraft, missiles, and projectiles in flight, in mixing regions of air and combustible material in reactive flows, in mixing layers near unstable material interfaces in laser fusion applications, and in environmental fluid dynamics of atmospheric storms and of fluid mixing in rivers and estuaries. Over the years, computational fluid dynamics (CFD) has been increasingly recognized as a vital approach in the design and understanding of practical systems. The complete set of flow equations to model multicomponent systems is generally difficult to solve and entire disciplines and scientific communities flourish solving subsets of these equations for particular applications. Modeling multicomponent flows with accurate mathematical models and inclusive physics is challenging as it crosses boundaries of many disciplines. Also, most CFD codes often ignore or use approximate models to represent these physical processes because of the complexity and computational hardware/software necessities they require. Advancements in the state of the art computational hardware and software

The work was sponsored by the U.S. Department of Energy under Contract Number DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation. The author also acknowledges the use of the Intel Paragon XP/S 150 computer, located in the Oak Ridge National Laboratory Center for Computational Sciences (CCS), funded by the Department of Energy's Mathematical, Information, and Computational Sciences (MICS) Division of the Office of Computational and Technology Research.

algorithms is now making it possible to perform realistic simulations of many fluid dynamics phenomena with more accurate physics. The availability of computer power has not only reached a room-full of several-thousands of tightly-connected processors with hundreds of GigaFLOPS (billions of floating-point operations per second), but also a high-speed connectivity of such setups, be it between rooms across the hall or the state borders.

We have been historically involved in supercomputing simulations of fusion plasmas [1–5], internal combustion engines [6–8], particle dynamics, and transport systems [9–11], and have also analyzed the performance of multicomponent systems on the parallel machines [12]. The purpose of our current efforts is to revisit our work in multicomponent flows and functional parallelism, combining our experience in all these areas to enhance the applicability and accuracy of our simulation codes in terms of physics, numerics, and computation. The components have been tested on the Intel parallel systems and show high levels of efficiency. The increase of computing power and emergence of the heterogeneous metacomputing environments is an indicative of an emerging capability to simulate such complex systems.

The developments in the study of internal combustion engines have recently produced numerical models such as KIVA-3 [13] that is being used as a major component for our code development. As reported earlier, it has now been parallelized by our efforts. KIVA-3 is a block-structured, multidimensional finite-difference combustion code; applicable to laminar or turbulent flows, subsonic or supersonic flows, and single-phase or dispersed two-phase flows. Another unique capability is its inclusion of spray dynamics.

Block-structured mesh offers the ability of representing the problem domain with multiple blocks as long as the connectivity between blocks is maintained properly. One can represent each part of the domain with a separate block and let the program patch them together through the connectivity arrays that describe who the neighbor points are in all directions (left, right, back, top, bottom, and front). The connectivity is defined through indirect addressing and this makes it no longer necessary to store the grid points in a particular order. The mesh in each block is made up of arbitrary hexahedrons (cells), the corner of which are vertices. Each block is independent and surrounded by ghost cells in all directions to handle the inflow/outflow boundary conditions. Also, cell-face boundary conditions for all six faces of a cell permit greater flexibility and simplification in the application of the boundary conditions. The use of ghost cells and cell-face boundary conditions make it possible to apply the same physics, numerics, and boundary conditions to the smallest units of the domain (cells or blocks) as well as to the whole domain. This generality forms a convenient foundation for a block-wise distribution of our computational model (described in the next section) on systems of independent processors.

In Section 2, we will briefly introduce the governing equations and computational volumes used for discretizing them. In Section 3, we will illustrate the grid and data structure and how that plays into our dynamic domain decomposition strategy. In Section 4, we will describe the spatial and temporal elements of our scalable approach, followed by a summary in Section 5.

2. COMPUTATIONAL MODELING

The fundamental equations to model a turbulent, radiative, and magnetized gas phase reactive flow are the continuum time-dependent equations for conservation of the total mass density ρ, the individual chemical species densities ρ_m, the momentum density $\rho\mathbf{u}$, and the internal energy density e^f. These equations may be written as [1,14–18]

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \dot{\rho}^s, \tag{1}$$

$$\frac{\partial \rho_m}{\partial t} + \nabla \cdot (\rho_m \mathbf{u}) = \nabla \cdot \left| \rho D \nabla \left(\frac{\rho_m}{\rho} \right) \right| + \dot{\rho}^c + \dot{\rho}^s \delta_{m,s}, \tag{2}$$

$$\frac{\partial}{\partial t}\left(\rho\mathbf{u}+\frac{1}{c^2}\mathbf{q}^R\right)+\nabla\cdot\left(\rho\mathbf{u}\mathbf{u}+\vec{\mathbf{P}}^R\right)$$
$$=-\nabla p-A_0\nabla\left(\frac{2}{3}\rho\kappa\right)+\nabla\cdot\sigma+\mathbf{F}^s+\rho\mathbf{g}+\frac{\mathbf{J}\times\mathbf{B}}{c}, \tag{3}$$

$$\frac{\partial}{\partial t}\left(e^f+e^R\right)+\nabla\cdot e^f\mathbf{u}$$
$$=-p\nabla\cdot\mathbf{u}+(1-A_0)\sigma:\nabla\mathbf{u}-\nabla\cdot\left(\mathbf{q}^{\mathbf{f}}+\mathbf{q}^{\mathbf{R}}\right)+A_0\rho\epsilon+\dot{Q}^c+\dot{Q}^s+\mathbf{J}\cdot\mathbf{E}, \tag{4}$$

where $\dot{\rho}^s$ and $\dot{\rho}^c$ are source terms due to sprays and chemistry, and $\delta_{m,s}=1$ when species m is the same as species of the spray droplet. $\mathbf{q}^R, \vec{\mathbf{P}}^R$, and e^R are radiation related quantities: heat flux, pressure tensor, and energy density, respectively. $\mathbf{J}\cdot\mathbf{E}$ is the Joule heating term and is equal to $\mathbf{E}'\cdot\mathbf{J}'$, the rate of Joulean dissipation in fluid frame, plus $\mathbf{u}\cdot((\mathbf{J}\times\mathbf{B})/c)$, the rate at which the force $(\mathbf{J}\times\mathbf{B})/c$ does work on the fluid. D is the diffusion coefficient, A_0 is the flow parameter (0-laminar, 1-turbulent), ϵ and κ are the turbulent kinetic energy and dissipation rate, and σ is the Newtonian viscous stress tensor consisting of the first and second coefficients of viscosity, μ and λ. Finally, $\mathbf{q}^f$ is the fluid heat flux vector as $\mathbf{q}^f=-K\nabla T-\rho D\sum_m h_m\nabla(\rho_m/\rho)$, with T as the temperature and h_m as the specific enthalpy of species m.

The solution of fluid density, momentum, and energy fields involve interaction terms between the fluid field and the others such as spray particles, turbulence, radiative transfer, chemical reactions, and electromagnetic fields. Consequently, one would need to solve governing equations for those fields as well, depending on their influence on the fluid motion. When turbulence is considered ($A=1$), two turbulence transport equations are solved for κ and ϵ. The standard $\kappa-\epsilon$ equations may be written as

$$\frac{\partial\rho\kappa}{\partial t}+\nabla\cdot(\rho\mathbf{u}\kappa)=-\frac{2}{3}\rho\kappa\nabla\cdot\mathbf{u}+\sigma:\nabla\mathbf{u}+\nabla\cdot\left|\left(\frac{\mu}{Pr_\kappa}\right)\nabla\kappa\right|-\rho\epsilon+\dot{W}^s \tag{5}$$

and

$$\frac{\partial\rho\epsilon}{\partial t}+\nabla\cdot(\rho\mathbf{u}\epsilon)$$
$$=-\left(\frac{2}{3}c_{\epsilon_1}-c\epsilon_3\right)\rho\epsilon\nabla\cdot\mathbf{u}+\nabla\cdot\left|\left(\frac{\mu}{Pr_\epsilon}\right)\nabla\epsilon\right|+\frac{\epsilon}{\kappa}\left|c_{\epsilon_1}\sigma:\nabla\mathbf{u}-c_{\epsilon_2}\rho\epsilon+c_s\dot{W}^s\right|, \tag{6}$$

where $\dot{W}^s$ is external source due to interaction with the spray, and $c_{\epsilon_1}, c_{\epsilon_2}, c_{\epsilon_3}, Pr_\kappa$, and Pr_ϵ are constants determined through experiments and theoretical assumptions. The assumption of $\kappa-\epsilon$ for turbulence modeling in combustion systems is being questioned frequently. The central assumptions in the $\kappa-\epsilon$ model are the isotropic Eddy viscosity concept and the gradient diffusion model. Also, a local equilibrium assumption is made, which is valid with limitations at very high Reynolds numbers. Turbulence persists as one of the major theoretical and computational problems of fluid dynamics. Computers have not been able to solve most turbulent flow problems from first principles, even though the fundamental set of equations are generally agreed to be adequate. In principle, the computational region must be big enough to include the largest scales, and the mesh spacing must be fine enough to resolve the smallest scales of interest. For our future work, we are interested in Large Eddy Simulations technique for turbulence modeling.

The chemical reactions are symbolized by $\sum_m a_{mr}\chi_m \Longleftrightarrow \sum_m b_{mr}\chi_m$, where χ_m represents one mole of species m, and a_{mr} and b_{mr} are integral stoichiometric coefficients for reaction r. These coefficients must satisfy $\sum_m(a_{mr}-b_{mr})W_m=0$ to conserve mass in chemical reactions. The database for chemical reactions (kinetic and equilibrium) has to accompany the model and we currently follow the data available in KIVA-3.

The interaction of gas with particles is a complicated problem and the interaction terms appear in most governing equations. In essence, one has to compute the particle density first and then

derive its moments to obtain the source terms that appear in the fluid equations given above. We deal with two kinds of particles: spray particles and photons. We follow the work in [14] for spray dynamics. The time evolution of f, the particle distribution function, is given by

$$\frac{\partial f}{\partial t} + \nabla_{\mathbf{x}} \cdot (f\mathbf{v}) + \nabla_{\mathbf{v}} \cdot (f\mathbf{F}) + \frac{\partial}{\partial r}(fR) + \frac{\partial}{\partial T_d}\left(f\dot{T}_d\right) + \frac{\partial}{\partial y}(f\dot{y}) + \frac{\partial}{\partial \dot{y}}(f\ddot{y}) = \dot{f}_{\text{coll}} + \dot{f}_{\text{bu}}, \quad (7)$$

where the quantities $\mathbf{F}$, R, $\dot{T}_d$, and $\ddot{y}$ are the time rates of change of spray droplet's velocity, radius, temperature, and oscillation velocity $\dot{y}$. The terms $\dot{f}_{\text{coll}}$ and $\dot{f}_{\text{bu}}$ are sources due to droplet collisions and breakups. As noted here, f has ten independent variables in addition to time. These variables are the three droplet position components $\mathbf{x}$, three velocity components $\mathbf{v}$, equilibrium radius r (the radius of the droplet if it were a complete sphere), temperature T_d, distortion from sphericity y, and the time rate of change $\dot{y}$. Droplets break up if y is larger than unity and coalesce if the collision impact parameter b is less than a critical value b_{rc}.

The state of the radiation field is found through the radiative transfer equation which is a mathematical statement of the conservation of photons driven from the linearized Boltzmann equation [16,19]

$$\left(\frac{1}{c}\frac{\partial}{\partial t} + \hat{\Omega} \cdot \nabla\right)\varphi\left(\mathbf{r}, t, \hat{\Omega}, \nu\right) = \eta\left(\mathbf{r}, t, \hat{\Omega}, \nu\right) - \chi\left(\mathbf{r}, t, \hat{\Omega}, \nu\right)\varphi\left(\mathbf{r}, t, \hat{\Omega}, \nu\right), \quad (8)$$

where φ is specific intensity, η and χ are called emissivity and extinction coefficients, and $\hat{\Omega}$ is the directional unit vector. Numerical solutions to the linear transport equation for photons (or neutrons) occupies a significant portion of computational efforts in a wide variety of problems within production and specialized code environments. Probabilistic methods offer a global solution such as the average reaction rate density, or the number of particles penetrating a medium, whereas deterministic methods offer a more specific solution including the spatial distribution of particle density. Our focus is on deterministic (S_N discrete ordinates) methods.

Radiation can be very important in engines, industrial burners as well as nuclear systems [15, 20,21], where temperatures reach many thousand degrees Kelvin. In the furnace region of coal fired boilers, radiative heat transfer may account up to 80% of the total heat transfer. In the engines, when there is sooting, the soot particles emit and absorb radiation and radiation from a sooty flame can remove enough heat to significantly change the buoyancy of the hot products. Our experience [4,5] with inertial confinement fusion have also shown a significant effect on the plasma hydrodynamics by radiative transfer. A streaming (anisotropic) radiation flow out of inner layers of materials or gas is likely to be reabsorbed and emitted with in the system when the system is comparable in size with the mean free path of the radiation. This heat release and absorption process has profound effects on the overall dynamics of the system. Discrete ordinates method is an effective and accurate way of capturing the angular distribution of radiation, with the exception of ray effects.

The space-time evolution of the electromagnetic fields for a medium moving with velocity $\mathbf{u}$ is derived from the Maxwell's equations, Ampere's Law, Faraday's Law, and Ohm's Law [22,23],

$$\begin{gathered} \frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B}) + \frac{c^2}{4\pi\sigma}\nabla^2\mathbf{B}, \\ \nabla \times \mathbf{B} = \frac{4\pi}{c}\mathbf{J}, \qquad \nabla \times \mathbf{E} = -\frac{1}{c}\frac{\partial \mathbf{B}}{\partial t}, \qquad \mathbf{J} = \sigma\left(\mathbf{E} + \frac{1}{c}\mathbf{u} \times \mathbf{B}\right), \end{gathered} \quad (9)$$

where σ is the electrical conductivity of the plasma. The first equation describes how the magnetic field lines are convected and diffused in the nonrelativistic and low-frequency plasma fluid. Others basically constitute a relation between the magnetic field $\mathbf{B}$, the beam density $\mathbf{J}$, and the electric field measured in the laboratory frame. The effects on the fluid, as seen in these governing

equations, are both electromagnetic and mechanical. The magnetic field creates a force on the fluid that is equivalent to a magnetic hydrostatic pressure ($B^2/8\pi$) via the $\mathbf{J} \times \mathbf{B}$ term in the momentum equation. There may be a need for other auxiliary equations or boundary conditions to solve the electromagnetic field, depending on the problem under consideration. In cases of spark-ignition in an engine or plasma channels in inertial confinement fusion reactors, either the current profile or the breakdown voltage, and spark characteristics and channel parameters (conductivity, geometry) need to be known for computing the current density $\mathbf{J}$. Previous models of spark ignition have been crude and based on empirical studies.

The sparking event in KIVA-3 is currently modeled through a single input energy parameter, lacking the details of spark ignition dynamics and its interaction with the fluid field and with electrode surface. Such estimates and the lack of details mostly come from the randomness in spark breakdown that precludes one from understanding the sparking process both qualitatively and quantitatively. Spark breakdown has been a random process with 20% to 40% uncertainties in deposited spark energy from cycle to cycle. A new spark-ignition control mechanism by Sauers [24] reduces this uncertainty down to less than 1%, permitting for the first time a systematic study of the combustion process and emissions with respect to the spark ignition energy. A model that eliminates cyclic variability due to spark breakdown would make it possible to study the influence of turbulence on the spark kernel and the kernel growth process with greater confidence. With a repeatable spark, the cyclic variability of the engine would now be exclusively due to the effect of turbulence and this might shed light even on the nature of engine turbulence.

The equations presented here are quite unique because of the degree of physics they involve. They have been written as a result of our past and current experience and of sweeping the literature in a variety of fields, including combustion, compressible flow, plasma physics, and radiation hydrodynamics. Usually, a subset of these is used by researchers depending on what the dominant component is. In the field of combustion and compressible flow, the effects of electromagnetic fields and radiative heat transfer are usually ignored and this certainly introduces poor predictability in the model when we compare it to the real world. It is our belief that the increase in computing power through parallel processing will enable researchers to construct more accurate computational models that are inclusive of most of the physical events.

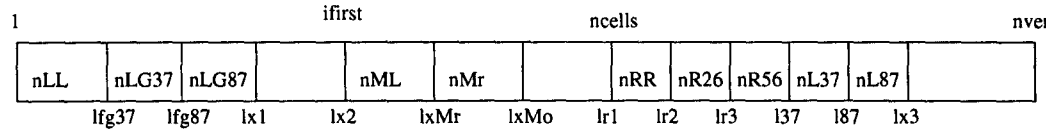
Most of the physical quantities in these equations are cell-averaged and are computed at cell centers via control volume approach. The difference equations over a control volume involves the use of cell-face variables and cell-face values are found as averages of cell-center values sharing the face in question. The momentum field is computed over the momentum cells that are centered at the vertex locations. Momentum cells occupy $1/8^{\text{th}}$ volume of each cell that share the vertex in question. The solution procedure involves computing cell-face velocities that require pressure contributions from both sides of the faces. Cell-face velocities are computed at cell faces via cell-face volumes centered around the face it involves. Overall, the numerical scheme uses three distinct volumes for discretizing the equations: regular cells, momentum cells, and cell-face cells (see Figure 3).

3. GRID DYNAMICS

The years of experience built in the CFD studies have produced very many numerical schemes and methods to solve fluid equations in different flow regimes, yet physical processes such as convection and diffusion often need to be treated in isolation of each other. Solving the flow field on a Lagrangian grid eliminates the convection term altogether. However, one still would need to account for convection if the final grid is not Lagrangian. Grid generation and dynamics is a basic part of the CFD modeling and requires additional attention in a distributed-memory implementation.

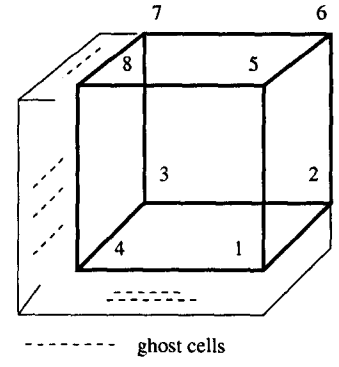
We start with a block-structured tensor-product grid made up of hexahedrons. Each hexahedron is an active cell with vertices located at the corners and is represented by its lower left-corner

vertex. The grid points can be Eulerian, Lagrangian, or move in an arbitrary manner described by the user. The data (grid locations, physical quantities) for such a tensor-product grid could be stored in 1D arrays by a single index that increases in an x-y-z order sweeping x-y planes one at a time. Data stored in this fashion will still maintain the structure of the grid, and neighborhood connectivity is defined through the x-y-z ordering of the cell index. A computational loop over the entire mesh will sweep the elements of 1D arrays from the start to the end including nonactive cells represented by the vertices on the boundaries. Since boundary vertices do not represent real cells, this might create a significant loss in execution time when there are many of such nonactive cells. This problem can be solved by storing the connectivity information into separate arrays for each direction and using indirect addressing through these arrays to identify the neighbor points in the grid [13]. Using indirect addressing and connectivity arrays, one can sort out the array elements in any way needed to increase the computational efficiency. With the exception of separating the active and nonactive cells, the grid does not need to be in any given order. Yet, in a parallel distributed implementation the communication needs require access to elements on the shared boundaries between processors, thus requiring at least a partial order within active and nonactive elements.



Number of sorted elements and group description:

nLL: f=0,fv=0 elements on the left; excluding front, bottom, top, back layers
nLG37: f=0, fv=0 elements on the left along 3-7 edge excluding the top layer
nLG87: f=0, fv=0 elements on the left along 8-7 edge , excluding front layer
nML: f=1, fv>0 elements on the left
nMr: f=1, fv>0 elements on the right
nRR: f=0, fv>0 elements on the right, excluding back and top layers
nR26: f=0, fv>0 elements on the right along 2-6 edge excluding the top layer
nR56: f=0, fv>0 elements on the right along 5-6 edge
nL37: f=0, fv>0 elements on the left along 3-7 edge excluding the top layer
nL87: f=0, fv>0 elements on the left along 8-7 edge
nRel: f=0, fv>0 elements: nRR+nR26+nR56



Communication Primitives:

Active <-> Ghost cell:	Vertex <-> Vertex copy:	Vertex <-> Vertex (gather):
sendLC(A(lx2),nMl)	sendLV(A(lx2),nMl,A(l37),nL37+nL87)	sendRV(A(lr1),nRel)
recvRC(A(lr1),nRR)	recvRV(A(lr1),nRel)	recvLV2(A(lx2),nMl,A(l37),nL37+nL87)
sendRC(A(lxMr),nMr)	sendRV(A(lr1),nRel)	sendLV(A(lx2),nMl,A(l37),nL37+nL87)
recvLC(A(1),nLL)	recvLV(A(lx2),nMl,A(l37),nL37+nL87)	recvRV(A(lr1),nRel)

Figure 1. The 3-D grid and 1-D data storage. Here, we show memory locations and the size of sorted groups as well as the elements subject to communication.

Figure 1 illustrates a 3-D tensor-product grid and our corresponding 1-D storage array sorted to keep not only active and nonactive cells apart, but also to match the boundary elements between two adjacent processors. The real physical domain is surrounded by ghost cells in all directions. The boundary vertices on the right, back, and top are real vertices, but they represent ghost cells that are not part of the physical domain. There are flags (F for cells, FV for vertices) associated with each cell and vertex to indicate if they are real or not. The flags take values of 1 or 0. After active and nonactive cells and vertices are sorted, the length of *do* loops will go from

ifirst to either *ncells* or *nverts*, instead of from 1 to *nverts*. Active cells are located between *ifirst* and *ncells* whereas active vertices are located between *ifirst* and *nverts*.

In a domain decomposition among parallel processors, two adjacent processor will need to communicate the values in cells and vertices on the shared boundary. The communication primitives would need to know where to access those elements in the memory and the list in Figure 1 demonstrates *send* and *receive* arguments between two processors (Left and Right). The chosen notation indicates the direction of the operation (L or R) and the type of information (C-cell, V-vertex). Each *send* is met with a *receive* operation.

The initial structured order of 1-D storage arrays is saved and used throughout the simulation whenever the grid moves or changed. For the engine problems, the grid needs to be dynamic due to piston motion. It is advisable to return to the original grid storage (before grid layers are added or deleted) to prevent deterioration in the grid due to nested sorting. It is also crucial to preserve a consistent interface between adjacent processors, and this can be best assured by going back to the original structured grid when layers of grid are either being added or deleted. The grid can be sorted again for enhanced *do loop* efficiency.

4. DISTRIBUTED APPROACH

4.1. Reactive and Turbulent Flow

The governing equations presented above all have a common form as

$$\frac{\partial Q}{\partial t} = -\nabla \cdot (Q\mathbf{u}) + \nabla \cdot D\nabla Q + S, \tag{10}$$

where the terms on the RHS are generally convection, diffusion, and source terms. These equations need to be discretized both in time and space. The numerical solution will mostly concentrate on the convection and diffusion processes, and these are the terms that will create spatial dependencies among processors when a distributed algorithm is considered. The temporal differencing is done in three different phases. Phase A is where most of the source terms are calculated, Phase B is where the diffusion terms are calculated, and Phase C is where the advection is computed. All these three phases apply to the same time interval, but one at a time built on top of each other. A variable implicitness strategy ($Q^{n+1} = \phi_D \cdot Q^{n+1} + (1 - \phi_D) \cdot Q^n$) is used to adapt the solution to the changing CFL condition. Here, ϕ_D is the implicitness variable and is computed at every time step to check the flow conditions. Implicit schemes generally pose difficulties for parallel computing due to the dependencies on the current values of the variable that is being calculated. However, the implicit solvers here are iterative ones and the physical variables on the RHS of the governing equations are already known (predictor), requiring no sequential dependencies between processors. However, the processors have to march together and make available to each other the necessary values at each iteration.

Spatial differences are done via the control volume, integrating the differential term in question over the volume of a regular or momentum cell. Most of the governing equations involve gradient and divergence terms as generalized in equation (10). Volume integrals are converted into surface area integrals using the divergence theorem and computation of such terms constitute the heart of the diffusion solvers in the code. Furthermore, the area integral over the surface of a cell can be written as a sum over the faces of the cell as

$$\int_V \nabla\phi \, dV = \int_S \phi \, d\mathbf{A} = \sum_\alpha \phi \mathbf{A}_\alpha$$

and

$$\int_V \nabla \cdot \nabla Q \, dV = \int_S \nabla Q \cdot d\mathbf{A} = \sum_\alpha (\nabla Q)_\alpha \cdot \mathbf{A}_\alpha,$$

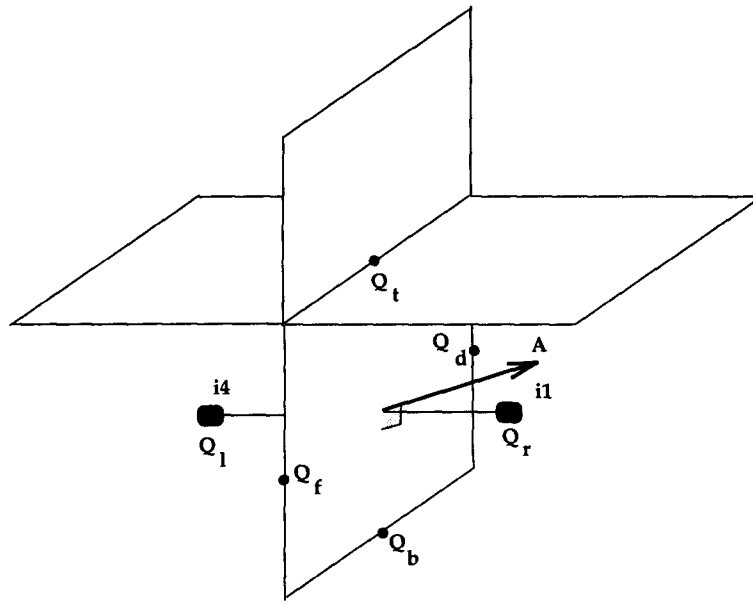


Figure 2. Cell-face averages used in the definition of $(\nabla Q)_\alpha$, The cell-face α is half-way between $i4$ and $i1$ cell centers here.

where α represents one of the six cell faces of a hexahedron. Evaluating the gradient on the surface is done as follows:

$$(\nabla Q)_\alpha \cdot \mathbf{A}_\alpha = \alpha_{lr}(Q_l - Q_r) + \alpha_{tb}(Q_t - Q_b) + \alpha_{fd}(Q_f - Q_d), \tag{11}$$

where r, l, t, b, f, and d indicates right, left, top, bottom, front, and back, respectively. Here, α_{lr}, α_{tb}, and α_{fd} are geometric factors for face α such that

$$\alpha_{lr}(\mathbf{x}_l - \mathbf{x}_r) + \alpha_{tb}(\mathbf{x}_t - \mathbf{x}_b) + \alpha_{fd}(\mathbf{x}_f - \mathbf{x}_d) = \mathbf{A}_\alpha, \tag{12}$$

where $\mathbf{x}_l$ and $\mathbf{x}_r$ are the center coordinates on either side of face α, and $\mathbf{x}_t$, $\mathbf{x}_b$, $\mathbf{x}_f$, and $\mathbf{x}_d$ are the centers of the four edges bounding face α. Here, also $Q_{r,l,t,b,f,d}$ are the average values computed at mid-points on the surface as shown in Figure 2. The surface integrals (or sums) involve evaluating physical quantities on the cell faces and the cell-face values are averages of the cell-center quantities sharing the face in question. This will be a source of interprocessor communications for cell faces on the boundary between adjacent processors.

Volume integrals over momentum cells are also converted to area integrals. As shown in Figure 3b, the momentum cells are constructed around vertices and involve eight regular cells that share a particular vertex. There are a total of 24 faces (β), three of each reside in one of the eight cells. In the computational loop over the vertices, each of the eight vertices contribute to the momentum via its three faces overlapping with the momentum cell in question. When the loop is finished the area integrals through all the faces are complete. The vertices on the boundary do not have as many β faces as the internal ones, unless the boundary is shared by other processors that have the remainder of the missing faces. Contributions from other processors will have to be gathered for physically internal but computationally external boundary vertex momentum cells.

Cell-face velocities require the pressure evaluation on both sides of the regular cell faces. As seen in Figure 3c, a cell-face volume is centered around the face in question with faces (γ) cutting through the cell centers on both sides. Again, the computation for cell-face velocities has to collect contributions from adjacent processors that share the face in question.

A block-wise decomposition (whether in one or multidirection) requires the processors share a common face. This in no way indicates that there is a duplication by the adjacent processors of

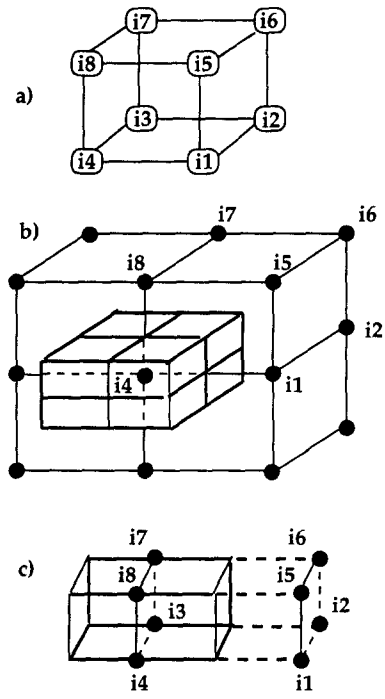


Figure 3. Computational cells: (a) $i4^{\text{th}}$ regular cell, (b) $i4^{\text{th}}$ momentum cell, (c) $i4^{\text{th}}$ left cell-face volume.

the vertices on the common faces. A vertex gets $1/8^{\text{th}}$ of its mass from each cell that has the vertex in common, and vertices on the processor boundary are only partially represented by each processor. Each processor applies the same rules (physics, numerics, and boundary conditions) to every computational cell, except some of the external boundary conditions are applied via the F flags. Care needs to be taken to treat physically internal but computationally external boundaries as internal by temporarily changing the flags to 1.0.

Advection of cell-center quantities require computing cell-face quantities and that in turn requires cell-center quantities and their derivatives on both sides of the face depending on the upwind conditions. Required information is gathered from other processors and put into ghost cells. Advection of momentum, however, is a somewhat different. One has not only to gather derivatives and variable values from vertices in the neighborhood, but also has to sum the contributions to account for all the momentum cell faces residing in multiple number of regular cells.

For a dynamic grid (such as a moving piston), the decomposition of blocks is parallel to the direction of motion for best possible load balancing and low communication/computation ratios. During this motion, the grid is decomposed, reorganized, and sorted again as done in the beginning of the simulation. There is need for processor communication to assure the removal of the grid at the same x-y planes. The interface between processors is assumed to have the same grid points on both sides to match the communication patterns when needed. A more general approach would eliminate this requirement by gathering into the ghost cells values computed by interpolations or extrapolations.

Since the access pattern for vertices and cells on the boundaries need to be known in advance for communication requirements, one needs to sort storage arrays, separating the elements on the left, right, and so on. Again, one would also have to assume that the boundary shared between neighbor processors has the same grid points on both sides to assure proper communication. The suggested sorting could be either done in the preprocessor or in the hydro program itself. The physical grid is surrounded by ghost cells in all six directions. These ghost cells may correspond

to the real cells of the adjacent blocks residing on other processors, thus creating a buffer area for storing boundary information that reside on the adjacent processors.

The testing of our flow algorithm is being done on the Intel Paragon and high parallel efficiencies (90%) have been measured for a simple baseline engine test case with 100 grid-points/processor in the direction of domain decomposition. The decomposition is done in x direction. The baseline engine problem has been successfully run up to 1024 processors on the Paragon without any major input/output or communication bottlenecks, considering that each processor uses multiple input/output files. The speedup on large number of nodes is problem-size dependent and one would need to keep the 100 grid-points/processor ratio of divisional work to operate on a high parallel efficiency. Though our tests so far involved relatively small problems, the true advantage of a distributed-memory implementation would be the ability to run large problems. A problem with a 6 Giga-Bytes of memory requirement (millions of mesh points) is certainly feasible on our 1024-node Paragon and future work will go into demonstrating that.

4.2. Particle Dynamics

The governing equations for spray dynamics are discretized over the same mesh system as we have described so far. The fluid flow is affected by the spray injection and the effects show up in the fluid equations, mostly in the source-term. Spray injection is local and may not occur in the domains of all the processors, causing a slight load-balancing problem. Particles cross cell faces and may end up changing processor domains. A parallel algorithm for particle destruction and creation across processor boundaries was developed based on our earlier work in a classical particle dynamics code (CMDT) [9,10]. CMDT has been well tested against similar codes in molecular dynamics and has run up to 400 million particles on our 1024-node Intel Paragon. It has been found to be a highly efficient algorithm (parallel efficiency greater than 90%) on the 1024-node GP Paragon and it is now ported to an MP Paragon with a speedup of 1.52. An Intel MP Paragon has nodes with 2 application processors (rather than one), letting scientists take advantage of not only a message-passing environment, but also the shared-memory parallelism within each node. Spray dynamics is a bit more complex than the classical Newtonian particle dynamics and the interaction between spray droplets involves breakup, evaporation as well as collisions and coalescenses.

4.3. Radiation Magnetohydrodynamics

The discrete ordinates solution of photon transport equation is very time-consuming because of the number of spatial, temporal, angular, and frequency variables involved. Discretization of equation (8) in multidimension can create enough floating-point operations to keep a supercomputer busy. The coupling of radiation transport with a flow field thus has always been prohibitive. Since the time-scale of radiative events is much smaller (on the order of nanoseconds) than the time-scale of the gas flow (on the order of microseconds), the solution of radiation field would need to be repeated many times during one time-step of fluid motion, making the coupling even more prohibitive. Among other things, the solution of radiation field depends on medium properties such as emissivity (η) and absorption (χ), which are functions of temperature, density, and specific internal energy. One can see two routes to a practical coupling from here: a loosely coupling of simultaneous simulations of flow and radiation fields, or an implicit solution of radiation field that would march with the flow solution time steps. Since gas conditions do not change much during a radiation time step, the coupling could be relaxed to an extent that information exchange only needs to take place on the order of flow time scale between flow and radiation solvers. The ratio of computational work between these solvers need to be proportional to the ratio of time scales for a well synchronized communication need. This would mean that the radiation solver should contain several orders of magnitude fewer floating-point operations (per radiation time step) than the flow solver (per flow time step). There have been a great

deal of research on parallel implementation of discrete ordinates methods. A recent survey by Azmy [25] looks at the history of multiprocessing for neutron transport methods. Parallelism has been exploited in angular, energy, and spatial domains. Our earlier work [11] with parallel discrete ordinates on Intel parallel machines yielded a high degree of parallelism (90%) for a 10 groups/processor energy decomposition, in which each processor is responsible for a portion (10 groups) of the energy (frequency) domain. The communication between different energy groups is almost negligible for photons (compared to neutrons) and this is responsible for such a high scalability.

The second route is to solve the radiation field via implicit schemes so that prohibitively small time steps are avoided. This implicitness, however, might create some dependencies in a parallel implementation unless we follow an iterative scheme similar to the one used for the reactive and turbulent flow. A domain decomposition would then be the strategy for parallelism. Each processor would apply the same solvers (reactive and turbulent flow, spray dynamics, radiation transport) to their separate domains.

We are expanding our earlier work [11] in parallel photon transport (discrete ordinates) to three dimension, considering not only energy decomposition but also domain decomposition that might prove useful in any both scenarios. An industry-standard 3-D discrete ordinates code (TORT) [26] is also part of our multidimensional parallel discrete ordinates tool-bag. One of the challenges is the handling of implicit solver for the radiation field, as it might create not only temporal but also angular dependencies for a domain decomposition. Angular boundary conditions for one processor require data from adjacent processors and this is likely to impose a sequential order in the parallelism. Efforts will go into implementing an iterative implicit scheme for radiation (as we do for the flow equations) to avoid this difficulty. Spatial dependencies are not of a prohibitive nature, however. Spatial discretization in discrete ordinates schemes usually involves Diamond Differencing (i.e., $\varphi_{n,i,j} = 1/2(\varphi_{n,i+1/2,j} + \varphi_{n,i-1/2,j})$, where n is angular, and i and j are spatial indices) to evaluate the flux on the cell faces. This is somehow similar to upwind differencing in terms of the depth of spatial dependencies for a distributed computation.

The solution of the magnetic diffusion, equation (9) involves a second spatial derivative that is, in general, discretized over a control volume in a *three-point central differencing* as $(\frac{d}{dx}[\frac{dB}{dx}])_i = 1/\Delta x(B_{i+1} - 2B_i + B_{i-1})$. This differencing will only require one-layer of spatial dependency between adjacent processors and suits well the domain decomposition strategy we described in Section 3. The temporal differencing of the magnetic field will require an implicit scheme to avoid small time steps. As we mentioned earlier, the radiation magnetohydrodynamics part of our parallel implementation is an ongoing effort and further progress is expected.

5. SUMMARY

A parallel block-wise decomposition finite-difference approach is described for multicomponent flows on distributed-memory machines. Spatial dependencies extend only one layer in each direction and the presence of ghost cells and cell-face boundary arrays suits a distributed-memory implementation well. There seems to be no dependency created through temporal differencing since variables are computed based on the quantities from the previous iteration or time step. Spatial differencing requires estimating variables and their gradients (diffusion terms) on the cell faces which leads to communication between adjacent processors sharing the cell-face in common. Momentum cells around the boundary vertices are split between processors, requiring each to compute their share of the vertex momentum. Advection involves fluxing through regular and momentum cell faces. Cell-face values that are found via upwind differencing require quantities and their derivatives on both sides of the face. Spray dynamics and chemistry require some communication and nonuniform distribution of particles can lead to slight load balancing problems. Particles cross processor boundaries and need to be created and destroyed. The grid points on the shared faces between processors need to have the same structure for predictable

communication patterns. Our distributed-memory algorithm is highly scalable and the speedup is problem-size dependent. Further work will go into extending the radiation and electromagnetic components to three dimension as well as applying our already developed Giga-Bytes and GigaFLOPS computing capability to large-scale problems.

REFERENCES

1. O. Yasar, A computational model for Z-pinch plasma channels, Ph.D. Thesis, University of Wisconsin-Madison, (December 1989).
2. O. Yasar and G.A. Moses, Explicit adaptive grid radiation magnetohydrodynamics, *J. Comput. Phys.* **100** (1), 38, (1992).
3. O. Yasar and G.A. Moses, R-MHD—An adaptive grid radiation magnetohydrodynamics computer code, *Comp. Phys. Comm.* **69**, 439, (1992).
4. O. Yasar, G.A. Moses and R.R. Peterson, Plasma channels for the LIBRA reactor designs, *Fusion Technology* **19**, 669, (1991).
5. O. Yasar and G.A. Moses, Z-pinch plasma channels for the LIBRA reactor design, *Nucl. Fusion* **31**, 273, (1991).
6. O. Yasar and C.J. Rutland, Parallelization of KIVA-II on the iPSC/860 Supercomputer, In *Parallel Computational Fluid Dynamics*, (Edited by R.B. Pelz *et al.)*, North-Holland, (1992).
7. O. Yasar and T. Zacharia, Distributed implementation of KIVA-3 on the Intel Paragon, In *Parallel Computational Fluid Dynamics*, (Edited by S. Taylor), North-Holland, (1995).
8. O. Yasar, T. Zacharia, A.A. Amsden, J.R. Baumgardner and R. Aggarwal, Implementation of KIVA-3 on distributed-memory MIMD computers, In *High-Performance Computing: Grand Challenges in Computer Simulation*, (Edited by A. Tentner), pp. 70–75, The Society for Computer Simulation, (1995).
9. Y. Deng, R.A. McCoy, R.B. Marr, R.F. Peierls and O. Yasar, Molecular dynamics on distributed-memory MIMD computers with load balancing, *Appl. Math. Lett.* **8** (3), 37–41, (1995).
10. Y. Deng, R.A. McCoy, R.B. Marr, R.F. Peierls and O. Yasar, Molecular dynamics for 400 million particles with short-range interactions, In *High Performance Computing: Grand Challenges in Computer Simulation*, (Edited by A. Tentner), pp. 95–100, The Society for Computer Simulation, (1995).
11. O. Yasar, G.A. Moses and T. Tautges, Radiation-magnetohydrodynamics of plasmas on parallel supercomputers, In *Proc. Frontiers of Massively Parallel Computation*, October 19–21, 1992, McLean, VA.
12. O. Yasar, T. Tautges, M. Corradini, and G.A. Moses, Multi-component flows in finite difference and control volume, In *High Performance Computing: Grand Challenges in Computer Simulation High Performance Computing*, (Edited by A. Tentner),pp. 66–69, The Society for Computer Simulation, (1993).
13. A.A. Amsden, KIVA-3: A KIVA program with block-structured mesh for complex feometries, Technical Report LA-12503-MS, Los Alamos National Laboratory, Los Alamos, NM, (1993).
14. A.A. Amsden, P.J. O'Rourke and T.D. Butler, KIVA-II: A computer program for chemically reactive flows with sprays, Technical Report LA-11560-MS, Los Alamos National Laboratory, Los Alamos, NM, (1989).
15. E.S. Oran and J.P. Boris, *Numerical Simulation of Reactive Flow*, Elsevier, New York, (1987).
16. D. Mihalas and B.W. Mihalas, *Foundations of Radiation Hydrodynamics*, Oxford University Press, (1984).
17. G.C. Pomraning, *The Equations of Radiation Hydrodynamics*, Pergamon Press, (1973).
18. W.F. Hughes and F.J. Young, *The Electromagnetodynamics of Fluids*, John Wiley, New York, (1966).
19. E.E. Lewis and W.F. Miller, Jr., *Computational Methods of Neutron Transport*, Wiley-Interscience, (1984).
20. M.P. Menguc, R. Viskanta and C.R. Ferguson, Multidimensional modeling of radiative heat transfer in diesel engines, SAE 850503.
21. J.J. Duderstadt and G.A. Moses, *Inertial Confinement Fusion*, John Wiley, (January 1982).
22. J.D. Jackson, *Classical Electrodynamics*, John Wiley & Sons, New York, (1975).
23. J.J. Watrous, Numerical and theoretical investigations of Z-pinch plasma channels for LIB inertial confinement fusion, Ph.D. Thesis, University of Wisconsin-Madison, (1986).
24. D.D. Sauers, J.W. Halliwell and C.W. Sohns, Energy delivery test measurement system using an automotive ignition coil for spark plug wire evaluation, Patent Disclosure ESID No. 1774XC, S-83,831.
25. Y.Y. Azmy, Multiprocessing for neutron diffusion and deterministic transport methods, *Progress in Nuclear Energy Journal* (to appear).
26. W.A. Rhoades and R.E. Flanery, In *1989 ANS Mathematics and Computation Conference*, No. 69, Santa Fe, NM, April 9–13; TORT is available at RSIC software repository at Oak Ridge National Laboratory.